Optimizing Sweep3D for the Cell Broadband Engine

Mike Lang

Performance and Architecture Lab (PAL)

CCS-1

- **Contributers:**
  - PAL Alumni
    - **Olaf Lubeck, Ram Srinivasan, Greg Johnson,**
  - PAL Team

    Kevin Barker, Kei Davis, Darren Kerbyson, Mike Lang, Scott Pakin, Jose Carlos Sancho Pitarch, and Adolfy Hoisie

LA-UR 08-2844

Talk Outline

- Intro (This Slide)
- Cell Architecture (brief review)
- Overview of Sweep3D (Why I'm not a physicist)
- Optimizations of Sweep3D
- Cell Messaging Layer (CML) Overview
- Performance and Roadrunner
- Conclusions and Futures

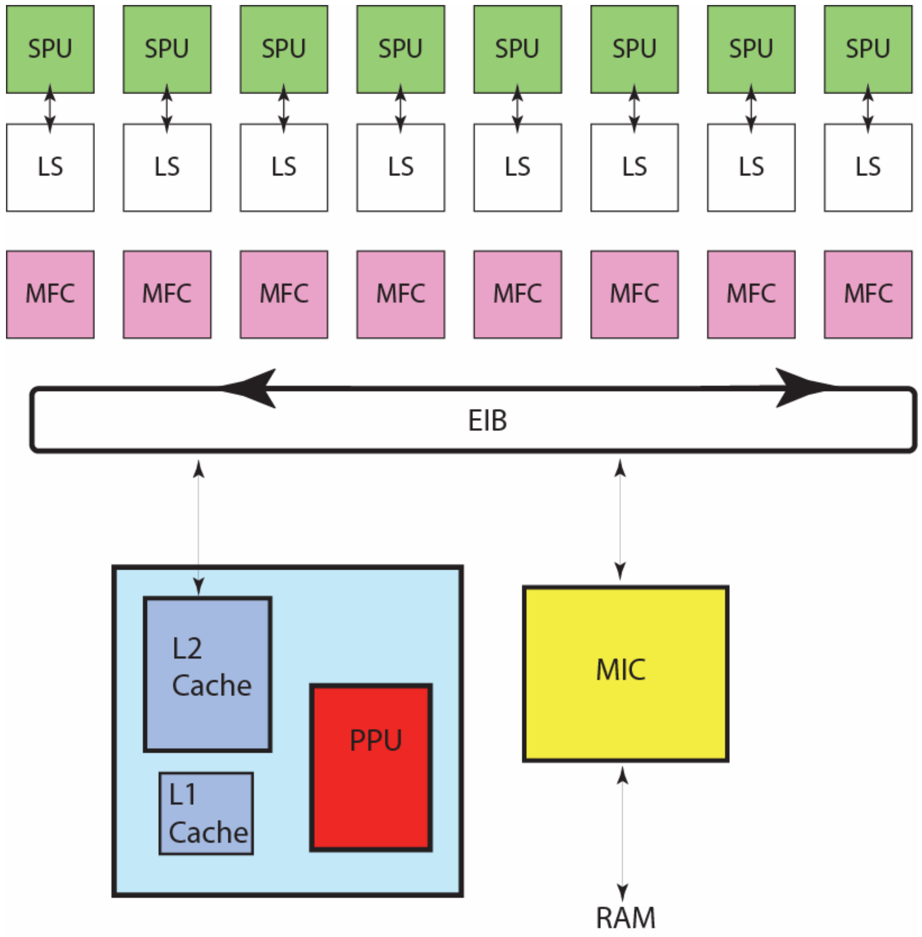
Cell Architecture (CBE)

Cell Broadband Engine
(Sony, Toshiba, IBM)

- PPE & SPE's (PPU & SPUs')?
- Advantages
  - » **High Peak Performance**
  - » **Fast Comms (EIB)**
- Disadvantages
  - » **Local store (manage your own memory)**
  - » **PPU**
  - » **Double precision slow**

PowerXCell 8i (edp, soma)

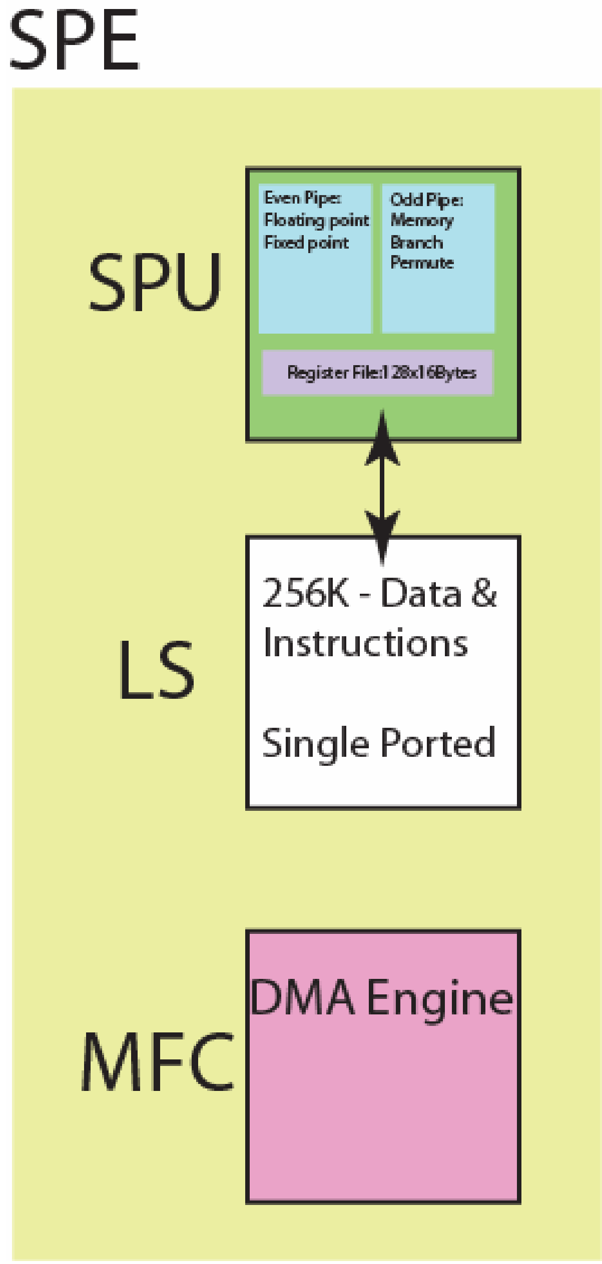
- Double Precision can be issued every cycle
- Swaps XDR memory for DDR2
  - » **Speed is very similar for 800Mhz – 25GB/s**
  - » **More memory (limited to 2GB with XDR)**
  - » **Less expensive**



Cell Architecture (SPE)

Really optimizing for this processor

- •SIMD (single instruction multiple data)
- •Two pipes (odd, even)
  - •2 instructions/cycle
- • Heterogeneous functional units
  - •7 units, in-order execution
- •Large number of registers (128)
- •Small Local store (256 KB)
- •High Speed DMA engine
  - •PPE Memory
  - •Other SPE's local stores
  - •Globally coherent memory space



Talk Outline

- Intro (This Slide)
- Cell Architecture (brief review)
- Overview of Sweep3D (Why I'm not a physicist)
- Optimizations of Sweep3D
- CML Overview
- Performance and Roadrunner
- Conclusions and Futures

Overview of Sweep3D

Physics

- **KBA Algorithm (Ken Koch, Randy Baker, and R. E. Alcouffe)**
- **Sweep3D solves a single group time-independent discrete ordinates (SN) neutron-transport problem deterministically.**
- **Sweep3D solves the particle transport equation, where the density distribution of the particles is the unknown.**

Decomposition

- **The sub-domain input size is specified in a 3-D with dimensions *I*, *J*, and *K*.**
- **The global data grid is decomposed in two dimensions across a logical 2-D processor array of size $n \times m$, giving a size $(I \times n) \times (J \times m) \times K$.**
- **The unit of work in Sweep3D is a block of the *K* dimension which is split into *K/MK* blocks. At most one block is computed on a processor in any one time-step.**

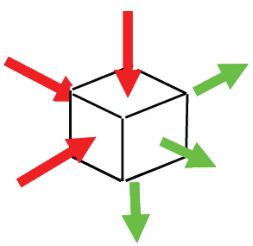
Note

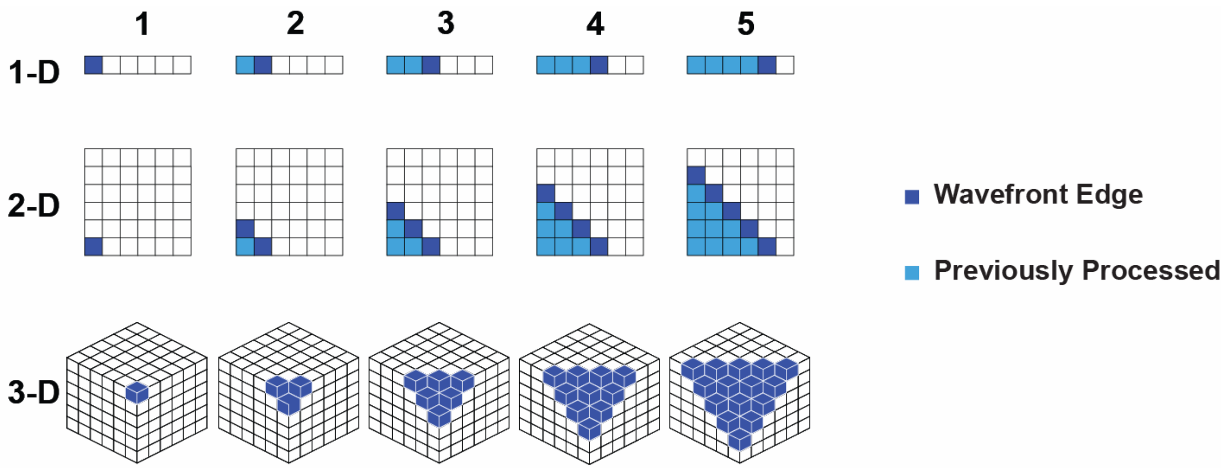
- **Blocking (MK) is used to achieve high parallel efficiency rather than to maximum cache utilization. (sweep pipeline)**
- **Deterministic, good for debugging, verifying.**

How does Sweep3D do its work

```
for octants
  for angles
    recv
      for k
        for j
          for i
            compute
    send
```

- **Sweep calculation**
  - Dependences between grid-points cause a diagonal wavefront that can be computed in parallel.
  - Inflows required before grid-points are processed
  - Outflows produced for downstream processors

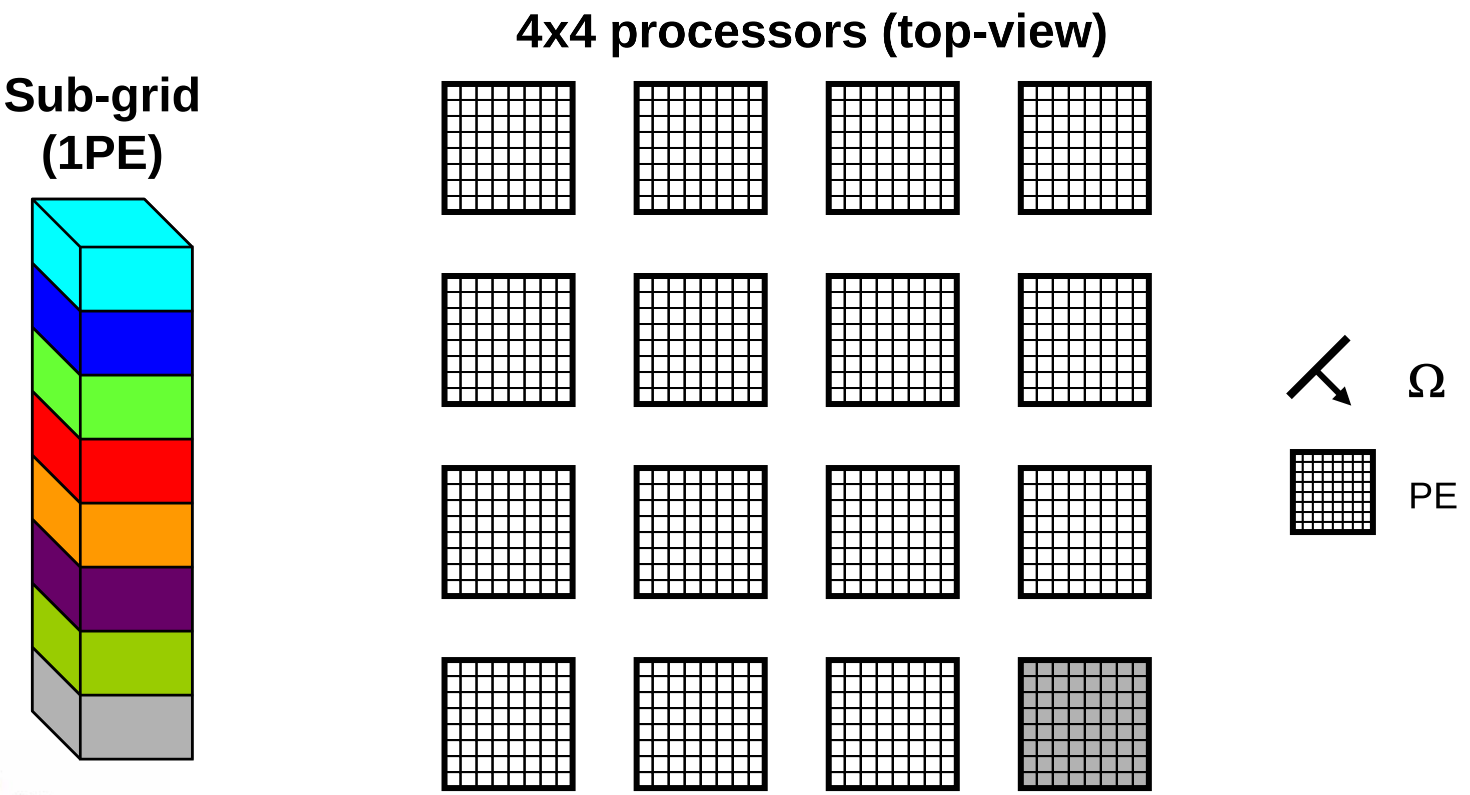




Wavefront starts from any corner of the cube

Sweep3D Wavefront Parallelization

- **Pipeline characteristic**
- **3-D grid is typically parallelized in only 2-D**
  – Blocking used to increase parallel efficiency

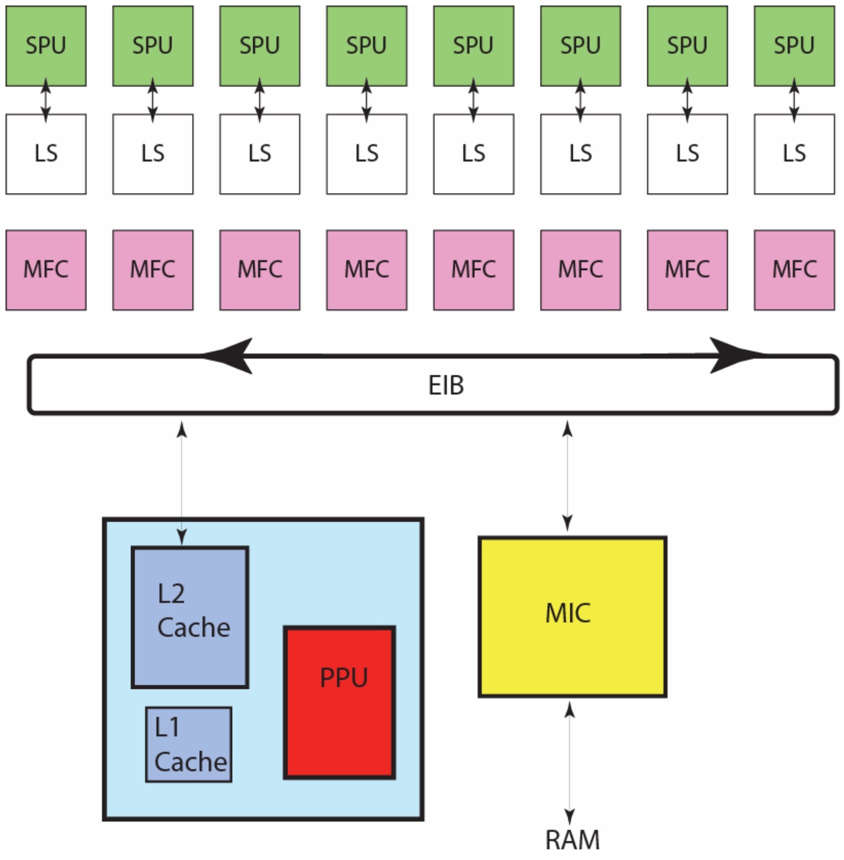


PAL Talk Outline

- Intro (This Slide)
- Cell Architecture (brief review)
- Overview of Sweep3D (Why I'm not a physicist)
- Optimizations of Sweep3D
- CML Overview
- Performance and Roadrunner
- Conclusions and Futures

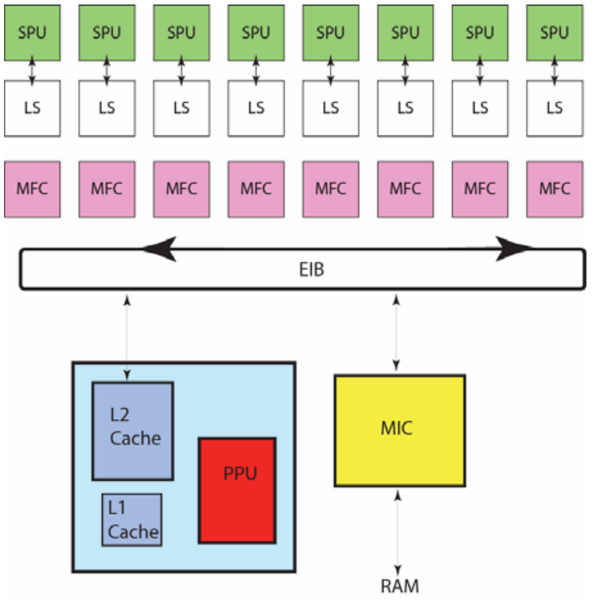
Mapping Sweep3D To Cell

- **Cell-Centric Implementation**
  - SPE = MPI Rank
- **Standard MPI decomposition with the SPE as the processing element (1 sub-grid per SPE)**
- **All computation is performed on SPE**
- **In-socket all comms over EIB**
- **Out-of-socket PPE forwards comms to other SPE's**
  - Requires MPI on SPE's



Optimizing Sweep3D

- **Balancing computation and communication (overlap)**
  - Computation time ~ subdomain volumes
  - Communication time ~ subdomain surface.
- **Minimizing memory traffic**
  - Decomposition blocked in K dimension to maximize amount of computation per DMA to main memory.
- **Use the architecture of the Cell**
  - EIB for local messages
  - Dual issue
  - SIMD




```
for octants
 for angles
  recv
   for k
    for j
     for i
      compute
  send
```

Taking advantage of SIMD

- SIMD (Single Instruction Multiple Data or vector)

How:

- Fixed the number of angles within an octant to multiples of six
- Re-ordered the nested loops so that the loop over angles was the innermost and the data was contiguous.
- processing of two of the six angles at a time utilizing SIMD instructions.
- This inner loop was then unrolled three times.
- In this way all six angles are efficiently computed on a single pass eliminating a loop.

Taking advantage of SIMD

- **Original:**

  for angles

  for k

  for j

  for i

- **Rearrange:**

  for k

  for j

  for i

  for angles

- **SIMD:**

  for k

  for j

  for i

  (A1,A2)

  (A3,A4)

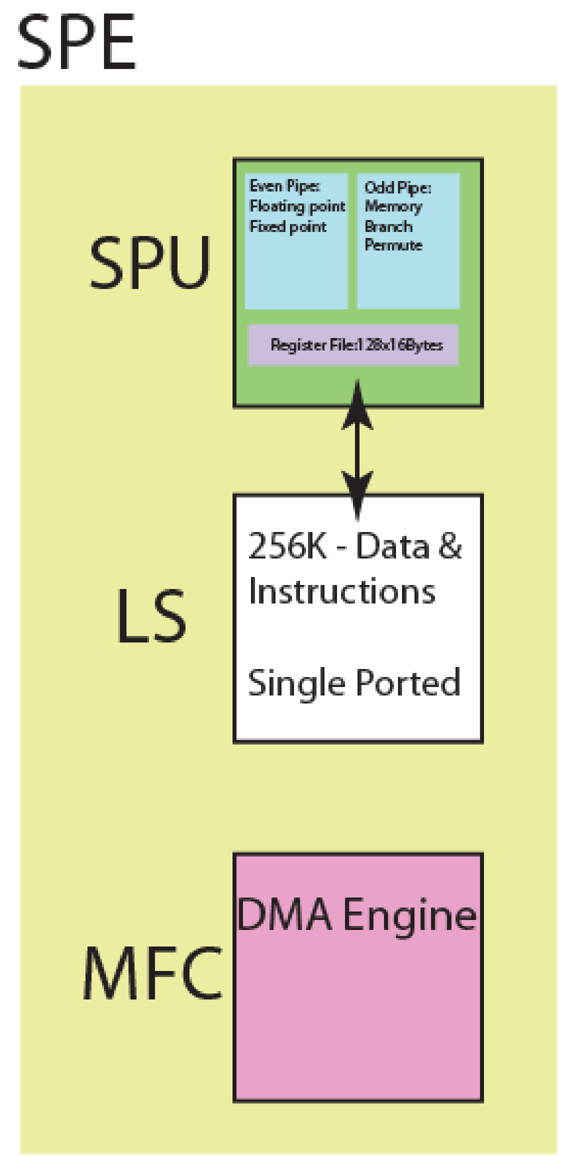
  (A5,A6)

Rearranging loops required data structure changes…

SPU SIMD intrinsics: spu_mul, spu_add, spu_madd …

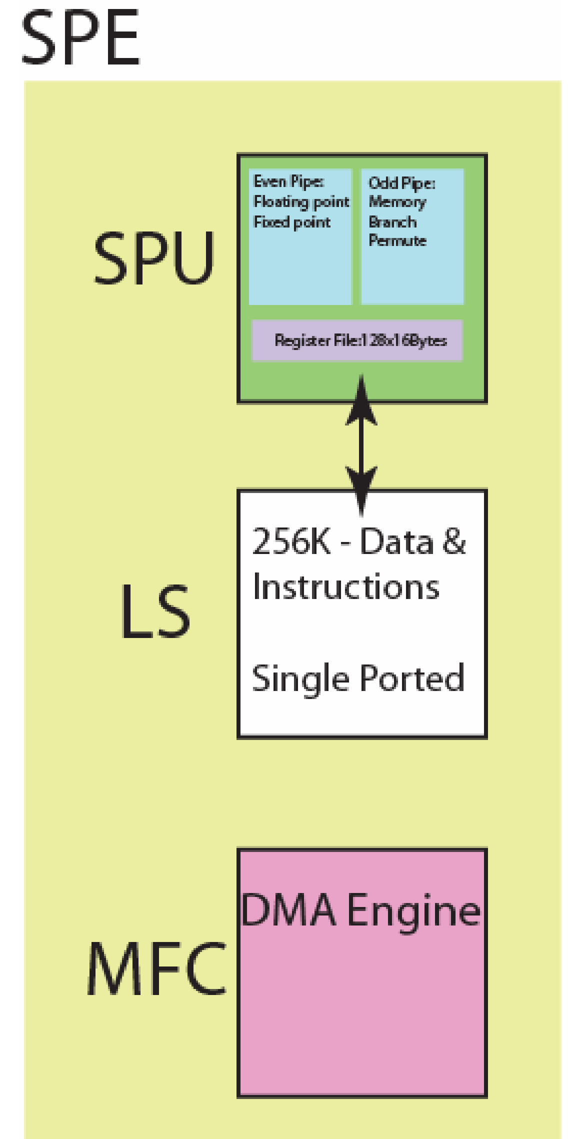
Taking Advantage of Dual Issue

- **The SPE is a dual-issue only if the correct instruction mix is available for the odd and even pipelines**
  - Interleave instructions
  - rearranging non-dependent code
  - unrolling and adding temporary variables, more instructions are available to fill the two pipes.
  - order of the instructions was carefully chosen to hide latencies.
  - Compute and loads/stores can be done concurrently
- **Acting like a compiler…**
  - Scheduling of instructions based on pipe
  - Scheduling of instructions based on latency
  - Unrolling loops



Taking Advantage of Dual Issue (example)

Pre-optimized	Even	Odd
Load A	NOP	Load A
Load B	R3=R3+1	Load B
Load C	R1=A*B	Load C
R1=A*B	R2=C*B	NOP
R2=C*B		
R3=R3+1		



Had to rearrange C-code looking at dependencies
Examine assembly
Time code

Instruction latencies

Instructions	Pipe	Latency
Arithmetic, logical compare, select	even	2
Shift, rotate, byte sum/diff/avg	even	4
Float Double Float (CBE +7)	even	6 6
16 bit integer multiply-accumulate	even	7
128 bit shift/rotate, shuffle	odd	4
Load, store, channel	odd	6
branch	odd	1-18

Other Optimizations

Problem in our SPU to SPU - MPI_Send

when copying a buffer to update neighbors

- **memcpy on SPU taking a lot of time**
  - Alignment of data is always an issue
    - » **DMA's not tolerant to alignment issues**
    - » **Default memcpy is alignment tolerant**
- **Re-wrote memcpy() to assume alignment**
  - Unrolled and uses vector assignment
- **In Sweep3D this was coping the MPI buffer to the data buffer.**

Initially “micro” MPI

- **Originally our MPI functions were tightly coupled to Sweep3D**
  - utilizing mutexes, arrays of mutually exclusive variables
  - atomic instructions to access mutexes
  - SPU to SPU DMA’s, polling mutexes for competition of communications
  - Hard to separate performance issues
  - Others were interested in using MPI on the SPU’s
- **Cell Messaging Layer (CML)**
  - Completely re-written as a independent library
    - **[2] Scott Pakin. Receiver-initiated Message Passing over RDMA Networks. In Proceedings of the 22nd IEEE International Parallel and Distributed Processing Symposium (IPDPS 2008), Miami, Florida, April 2008.**
  - available externally and GPL’ed http://www.c3.lanl.gov/~pakin/software/cellmessaging/

Overview of Cell Messaging Layer (CML)

- **Easy to use**
- **Minimal MPI layer for SPU's**
  - Each SPU has a unique MPI rank
  - Each SPU can exchange messages with any other rank
- **Optimized for speed, code size**
- **Minimal use of the PPE (slow processor)**
- **CML (currently)**
  - Single Socket (8 SPU's)
  - Blade (16 SPU's) over EIB connection
  - Cell cluster (AAIS)

CML MPI functions implemented

- MPI_Abort()
- MPI_Allreduce()
- MPI_Barrier()
- MPI_Bcast()
- **MPI_Comm_get_attr()**
- MPI_Comm_rank()
- MPI_Comm_size()
- MPI_Finalize()
- MPI_Init()
- MPI_Recv()
- MPI_Reduce()
- MPI_Send()
- MPI_Wtime()
- MPI_Wtick()

CML Bonus:

The Cell Messaging Layer supports a remote procedure call (RPC) mechanism.

This enables a SPE to invoke functions on the PPE and receive the results.

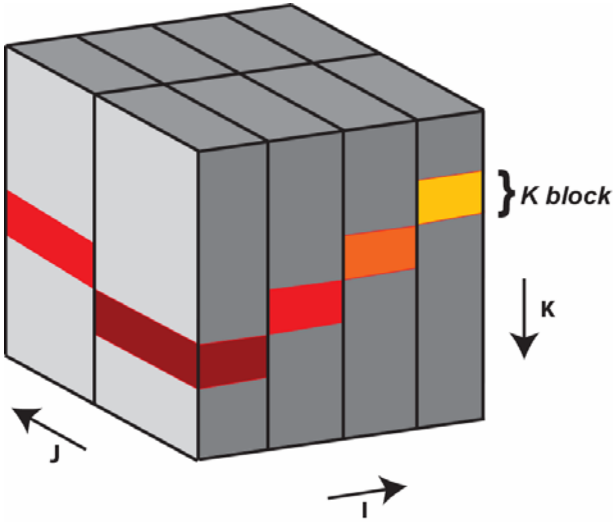
(Just happens to be the MPI calls sweep3D needs)

CML limitations

- **Subset of MPI**
- **Message tags are ignored.**
- **MPI_ANY_SOURCE is not supported.**
- **MPI_COMM_WORLD is the only valid communicator.**
- **MPI_Send() synchronizes with the receivers**
  - MPI's MPI_Ssend().
- **MPI_Reduce() and MPI_Allreduce()**
  - support only a few datatypes and operations on those datatypes.
- **MPI_Wtime() wraps**
  - The cycle counter used by the CBE wraps every 232 cycles 5 mins
- **MPI_Comm_get_attr(), limited**
  - accepts MPI_WTIME_IS_GLOBAL, MPI_TAG_UB, and MPI_CML_LOCAL_NEIGHBORS

Summary of Sweep3d CBE Optimizations

- **Cell-Centric Implementation, SPU=MPI-rank**
- **Balancing computation and communication**
  - Minimize DMA's to main memory
  - Overlap with compute
- **Vectorizing for the SIMD SPU instructions**
- **Dual Issue - Issues**
  - Instructions scheduled with respect to odd/even SPE pipelines and latencies.
- **Moved to CML**



Talk Outline

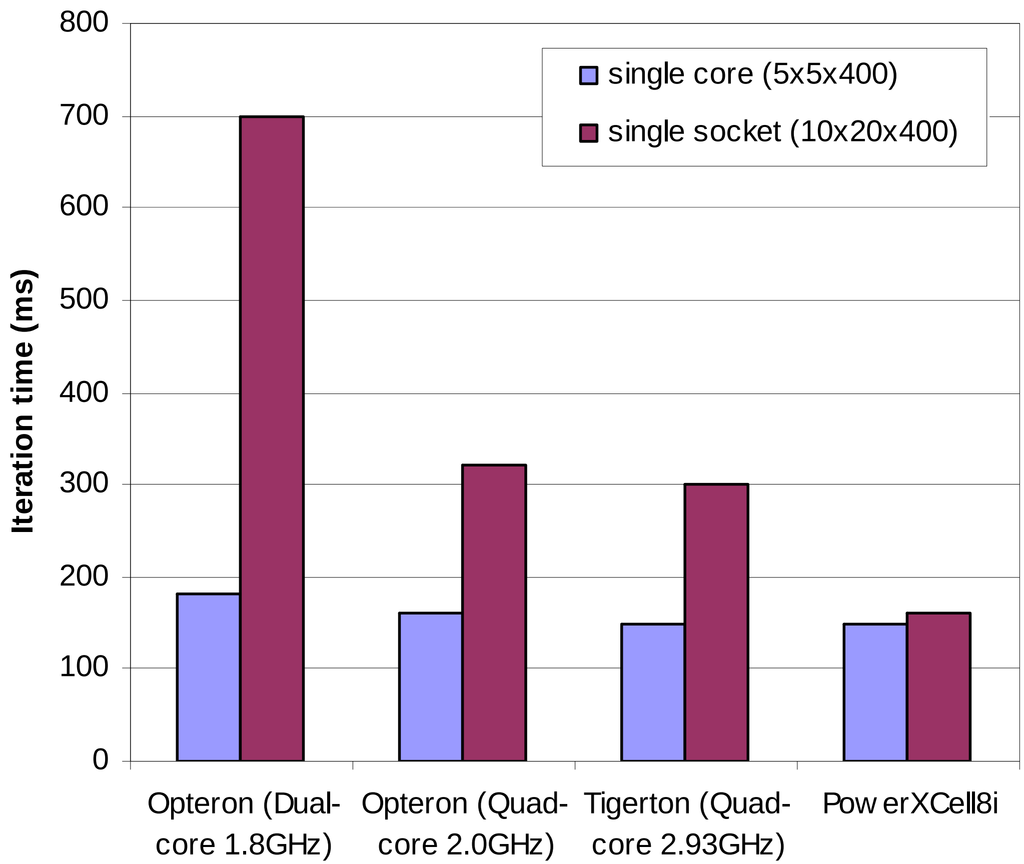
- **Intro (This Slide)**
- **Cell Architecture (brief review)**
- **Overview of Sweep3D (Why I'm not a physicist)**
- **Optimizations of Sweep3D**
  - CML Overview
- **Performance and Roadrunner**
- **Conclusions and Futures**

Performance was it worth it? compared to a socket

9x comparing PowerXCell 8i socket to a 2GHz Opteron core.
1 SPU ~= 1 Conventional Core

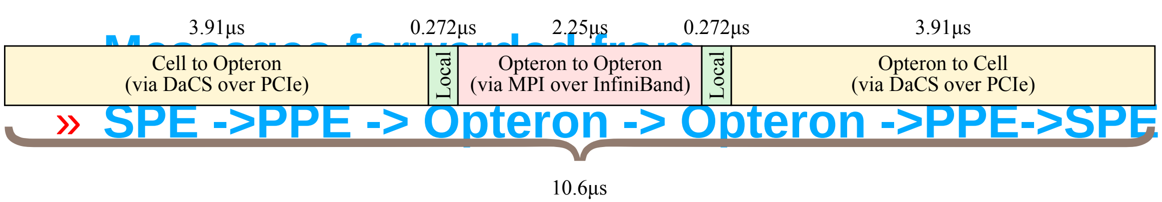
- Single core compares
  - Single SPU to Single core
  - Each core does same work

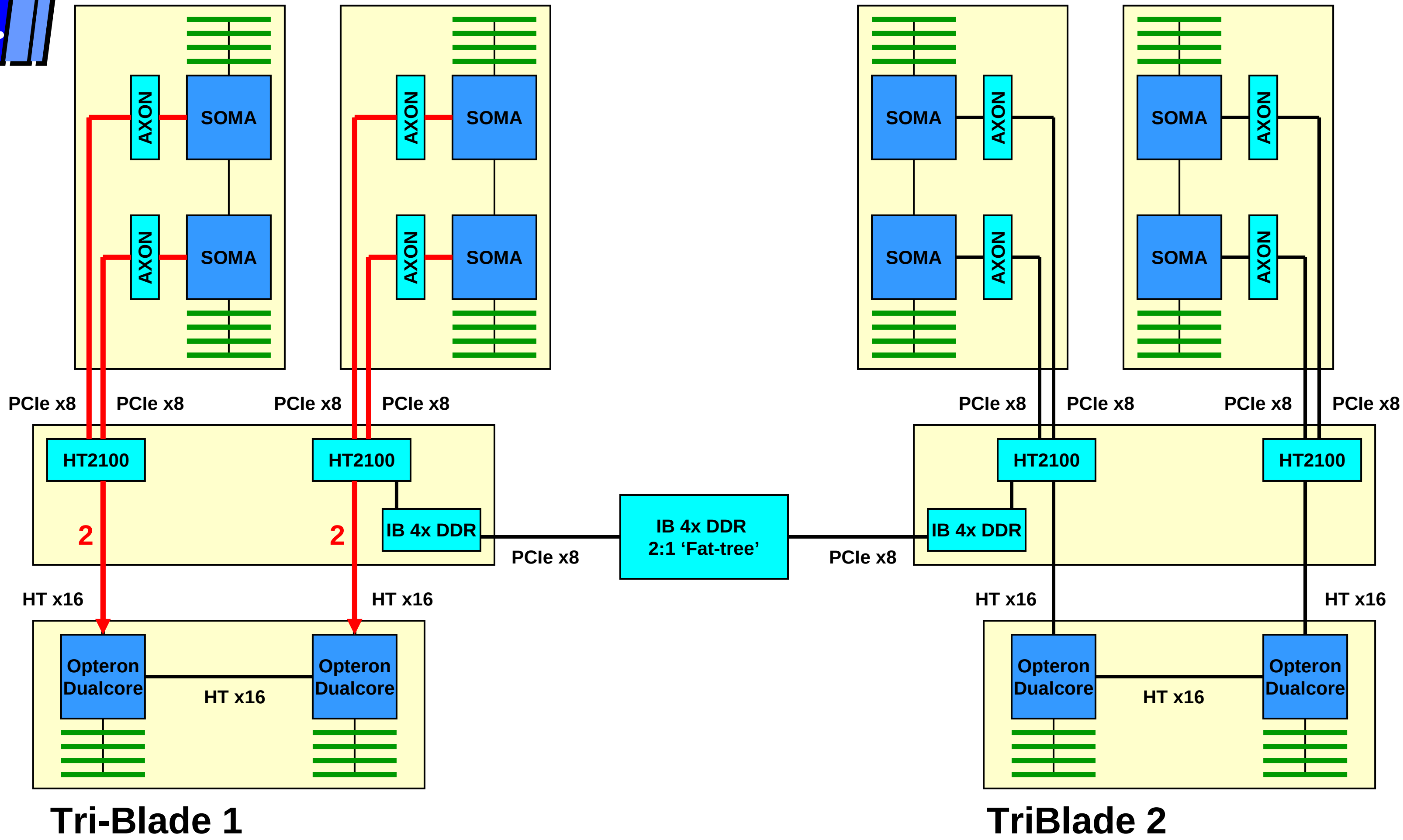
- Single Socket compares
  - 8 SPU's to 2 or 4 cores
  - Each socket does same work



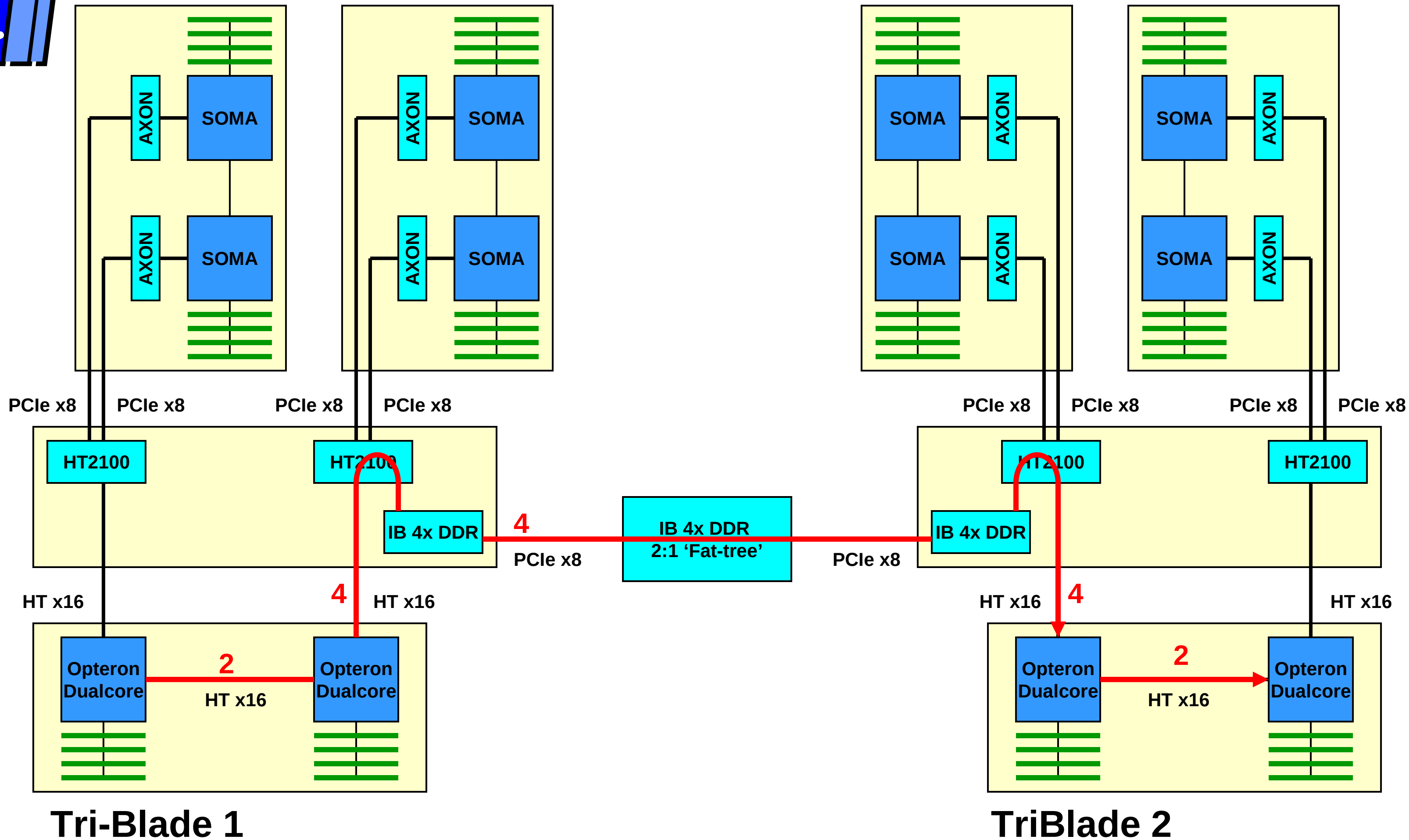
But what about Roadrunner?

- **So far a results for a single Cell Blade**
  - (up to16 SPES)
- **CML previously supported Cells interconnected with a network (AAIS)**
  - Forwards MPI from PPE to PPE
- **Roadrunner is different**
  - no direct communication path (only through Opteron)
- **To port Sweep3D we are modifying CML to be hybrid**
  - Messages forwarded from
    - SPE ->PPE -> Opteron -> Opteron ->PPE->SPE



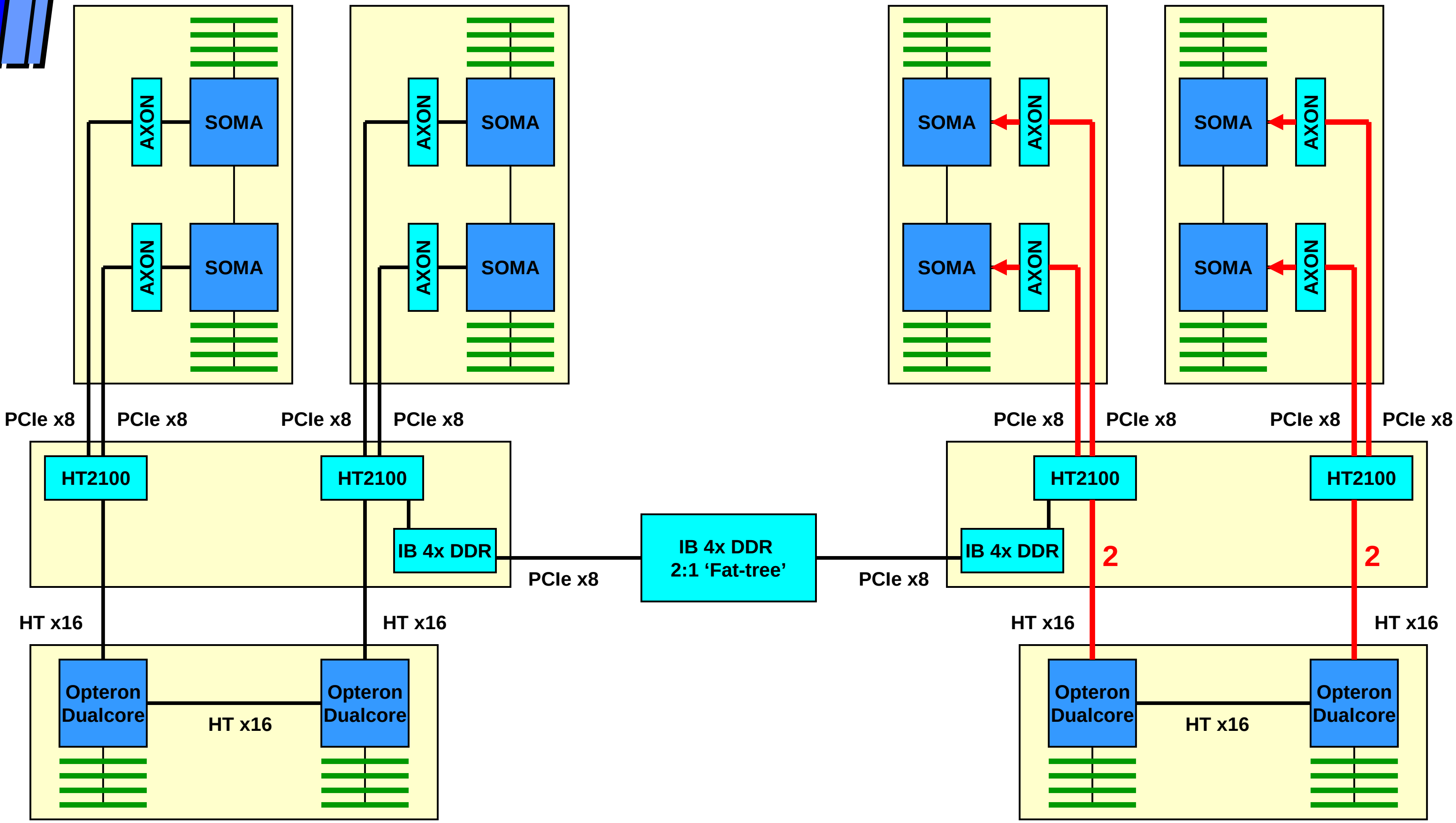


1) Cells (TriB 1) -> Opterons (TriB 1)
2) Opterons (TriB 1) -> Opterons (TriB 2)
3) Opterons (TriB 2) -> Cells (TriB 2)



1) Cells (TriB 1) -> Opterons (TriB 1)
2) Opterons (TriB 1) -> Opterons (TriB 2)
3) Opterons (TriB 2) -> Cells (TriB 2)

PAL



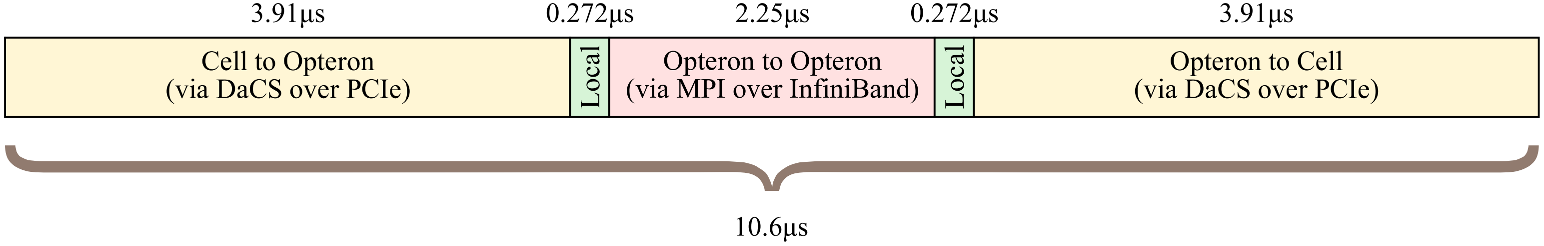
Tri-Blade 1 TriBlade 2

1) Cells (TriB 1) -> Opterons (TriB 1)
2) Opterons (TriB 1) -> Opterons (TriB 2)
3) Opterons (TriB 2) -> Cells (TriB 2)

Sweep3D Workload Model

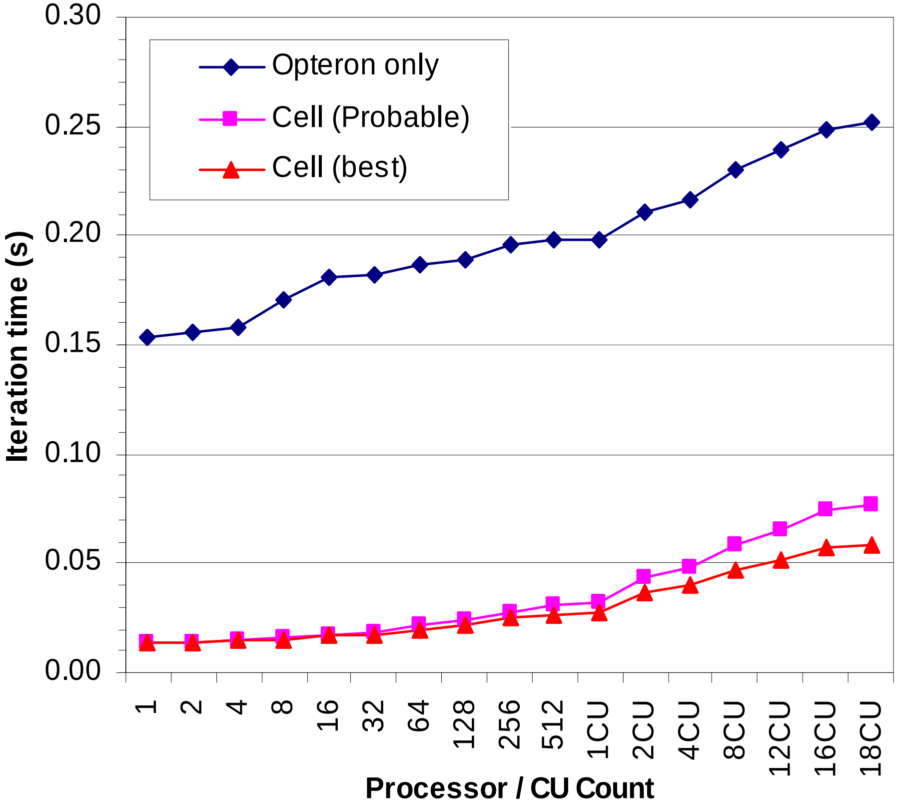
March 19:"Overview of Modeling, Performance, and Results," Darren Kerbyson

- **Message characteristics**
  - Fine-grained communications:
    - **2 messages sent per SPE per block per cycle**
    - **Sizes depend on block size, 240B -> 4,800B (typical)**
- **Performance Model validated on all large-scale systems**
- **Model adapted to reflect additional Cell->AMD communications**



Current Status

- Expected performance is shown by modeling:
- Just got it running, no performance data yet.



Conclusions - futures

- **Conclusion**
  - Tools are getting better
  - Compilers are getting better
  - Libraries are more stable

- **Futures:**
  - Other optimizations (message aggregation)
  - Will this work of other codes?
    - » **Overlays, Scale**
  - Initial investigation started for Partisn in collaboration with CCS-2
    - » **Large scale FORTAN code**
    - » **Sweep3D is one of the solvers in Partisn**

Thanks and Questions !!

- **Slides will be available on roadrunner web site**
  - www.lanl.gov/roadrunner

PAL Roadrunner Technical Seminar Series

March 13: "Roadrunner Platform Overview," Ken Koch, CCS-DO.

March 18: "Overview of Applications, Results, and Programming," John Turner, CCS-2

March 19: "Overview of Modeling, Performance, and Results," Darren Kerbyson, CCS-1

April 10: "Application 1: SPaSM," Sriram Swaminarayan, CCS-2

April 22: "Application 2: VPIC," Ben Bergen, CCS-2

April 23: "Application 3: SWEEP," Mike Lang, CCS-1

****April 24: "Application 4: Milagro 1," Tim Kelley, CCS-2****

May 6: "Application 5: Milagro II," Paul Henning, CCS-2

May 8: "Application 6: DNS," Jamal Mohd-Yusof, CCS-2

May 29: "Panel Discussion: Hybrid Computing Programming Models,“

June 3: "Panel Discussion: Future Platforms,"